MIT-CTP 4279

Component mixers and a hardness result for counterfeiting quantum money

Andrew Lutomirski

Friday, July 1, 2011

Abstract

In this paper we give the first proof that, under reasonable assumptions, a problem related to counterfeiting quantum money from knots [1] is hard. Along the way, we introduce the concept of a component mixer, define three new classical query problems and associated complexity classes related to graph isomorphism and group membership, and conjecture an oracle separating QCMA from QMA.

1 Introduction

Quantum money from knots is a cryptographic protocol in which a mint produces a quantum state $|\$_\ell\rangle$. Anyone can verify that the state came from the mint, and hopefully it is intractable to copy the state. The quantum state is described by a set S which is partitioned into components. (In practice S is a set of knot diagrams with bounded complexity and the components are the sets of knots with the same Alexander polynomial.)

We hope to prove the security of an abstracted version of the protocol in which adversaries have only black-box access to an idealized version of knot-theoretic operations [2]. In this abstracted version, the quantum state is a superposition of n-bit strings (n is a security parameter chosen by the mint). All of those strings come from a large set S, and the state $|\$_\ell\rangle = \sum_{x \in S_\ell} |x\rangle$ is the uniform superposition of strings in the ℓ^{th} component.

All parties (the mint, honest users of money, and any adversaries) have access to two black-box operations. They have access to a "component mixer" (defined below) that invertibly maps any string to a new almost uniformly random string in the same component. They also have access to a labeling function that determines which component any element is in. (In the concrete scheme, the component mixer does not fully mix within the components. We ignore that issue here.)

To prove hardness results related to counterfeiting quantum money, we need an appropriate computational assumption. We find this assumption in a new class of query problems based on component mixers.

All of these problems involve a large set that is partitioned into components. An algorithm must use black-box queries to a component mixer to answer questions about the components. The algorithm is not given access to a labeling function. The SAME COMPONENT problem is: are two given elements in the same component? The MULTIPLE COMPONENTS problem is: is there more than one component (as opposed to just one)? If we promise that either there is only one component or no component contains a majority of the set, then the MULTIPLE COMPONENTS problem becomes MULTIPLE BALANCED COMPONENTS. Finally, on a quantum computer, the COMPONENT SUPERPOSITION problem is to prepare the uniform superposition of all elements in

The class...	...is in...	...and oracle-separated from
SCP	NP, SZK	co-MA, hopefully BQP
MCP	QMA, $\mathrm{NP}^{\text{co-NP}}$, $\mathrm{NP}^{\text{co-SCP}}$	BQP, hopefully QCMA
MBCP	MCP, AM, co-AM, SZK, $\mathrm{BPP}^{\mathrm{SCP}}$	hopefully QCMA

Table 1: Component problems have a home in the complexity zoo.

a component given as input one element in that component. (The classical analog, producing a uniformly random sample from a component, is easy by assumption.)

These types of questions are natural abstractions of graph isomorphism and group membership. Graph isomorphism is the problem of deciding whether two graphs are equivalent up to a permutation. The complexity of graph isomorphism is unknown on both classical and quantum computers. Group membership, on the other hand, is the problem of deciding whether an element of some large group is in a particular subgroup of that group. The subgroup is specified by its generators, and the group structure is given either as a black box or in some explicit form such as a matrix representation. In the black-box setting, group membership is hard on classical computers [3] but has unknown complexity on quantum computers.

For graph isomorphism, the big set would be the set of all graphs of a given size and the components are isomorphism classes of graphs. The component mixer permutes the vertices of a graph. Testing whether two graphs are isomorphic reduces to the SAME COMPONENT problem.

For group membership, the big set would be a large group and the components would be cosets of a subgroup that is described only by its generators. The component mixer would multiply by an element of the subgroup. The group membership problem reduces to an instance of SAME COMPONENT: testing whether a given element is in the same component as the identity. The MULTIPLE BALANCED COMPONENTS problem would determine whether the given generators generate the entire group or a proper subgroup.

Each of these query problems naturally defines a complexity class. SCP, MCP, and MBCP are the sets of languages that are polynomial-time reducible to the SAME COMPONENT problem, MULTIPLE COMPONENTS problem and MULTIPLE BALANCED COMPONENTS problem. We relate all three classes to commonly-used complexity classes. Our results are summarized in the table above.

These problems and classes are immediately interesting for two reasons. First, if SAME COMPONENT is hard on a quantum computer, then we have evidence for the security of a quantum money protocol [1]. Second, MCP and MBCP are candidates for a classical oracle separation between QCMA and QMA. (Group membership does not work directly because it has too much structure [4].)

2 Definitions

Throughout this paper, we use some basic terminology. We say that $a \in_R B$ if a is a uniform random sample from B. A function $f : \mathbb{N} \to \mathbb{R}$ is negligible if for all y there exists N_y such that $f(x) < x^{-y}$ for all $x > N_y$. Intuitively, negligible functions go to zero faster than the reciprocal of any polynomial. Finally, the total variation distance between two distributions with probability density functions p and q over a set D is

$$\frac{1}{2}\sum_e |p(e) - q(e)| = \sup_{A \subseteq D} |p(A) - q(A)|.$$

The total variation distance is sometimes referred to as the statistical difference, and it is analogous to the trace distance between density matrices of mixed quantum states.

All of the problems we consider are questions about a large set S. For consistency in defining the size of the problems, we take n to be the number of bits used to represent an element of S. The set S is partitioned into components, and access to the components is given through a family of invertible maps that takes an element of any component of S to a new element of the same component. The maps constitute a component mixer if a uniform random choice of the map produces a uniformly random output.

Definition 1. A family of one-to-one maps $\{M_i\}$ is a *component mixer* on a partition $\{S_1, \ldots, S_c\}$ of a set S if:

- The set S is a subset of n-bit strings.
- The family is indexed by a label i from a set Ind_M, and each i can be encoded in $O(\mathrm{poly}(n))$ bits.
- The functions $\{M_i\}$ do not mix between components. That is, for all i and a, if $x \in S_a$ then $M_i(x) \in S_a$ as well.
- The functions $\{M_i\}$ instantly mix within each component. That is, for all a and $x \in S_a$, if $i \in_R \mathrm{Ind}_M$, then the total variation distance between $M_i(x)$ and a uniform sample from S_a is no more than 2^{-n-2}.

The last condition is often easy to satisfy directly. For graph isomorphism, if Ind_M is the set of permutations, M_i can apply the permutation i to a graph. For group membership, if Ind_M is a sequence of coin flips that can generate a nearly uniform sample from the subgroup (e.g. a straight-line program as in [5]), then M_i can multiply by the element of the subgroup implied by the coin flips. In general, given a Markov chain over S that does not mix between components but mixes rapidly over each component, each step of which consists of choosing uniform random sample from a set of invertible rules and applying that rule, then iterating that Markov chain to amplify its spectral gap will give a component mixer.

In graph isomorphism, generating a random graph, testing whether some encoding of a graph is valid, and generating a random permutation to apply to the vertices are all easy. Similarly, in group membership, generating a random element of the whole group (as opposed to the subgroup) is easy, as is generating a random element of the subgroup. When we abstract these problems to component mixer problems, we want the corresponding operations to be easy as well. This leads to our definition of query access to a component mixer.

Definition 2. An algorithm has query access to a component mixer $\{M_i\}$ if the algorithm can do each of the following operations in one query with failure probability no more than 2^{-n}.

- Test an n-bit string for membership in S.
- Generate an uniform random sample from S.
- Test a string for membership in Ind_M.
- Generate an uniform random sample from Ind_M.
- Given $s \in S$ and $i \in \mathrm{Ind}_M$, compute $M_i(s)$.

- Given $s \in S$ and $i \in \mathrm{Ind}_M$, compute $M_i^{-1}(s)$.

If we are considering *quantum* algorithms, we want to give the algorithm some quantum power. For example, in graph isomorphism, generating a uniform quantum superposition of *all* graphs is easy, as is generating a uniform quantum superposition of all members of the permutation group [6]. We give quantum component algorithms the equivalent powers.

Definition 3. An algorithm has quantum query access to a component mixer $\{M_i\}$ if the algorithm can do each of the following operations coherently in one query with failure probability no more than 2^{-n}:

- Test an n-bit string for membership in S.
- Generate the state $\sum_{s \in S} |s\rangle$ or measure the projector onto that state.
- Test a string for membership in Ind_M.
- Generate the state $\sum_{i \in \mathrm{Ind}_M} |i\rangle$ or measure the projector onto that state.
- Compute the "controlled-M" operator, abbreviated CM. CM takes three registers as input: the first is the number -1, 0, or $+1$, the second is a string i, and the third is an n-bit string s. On input $|\alpha, i, s\rangle$, $\mathrm{CM}\,|\alpha, i, s\rangle = |\alpha, i, M_i^{\alpha}(s)\rangle$ if $i \in \mathrm{Ind}_M$ and $s \in S$; otherwise $\mathrm{CM}\,|\alpha, i, s\rangle = |\alpha, i, s\rangle$.

As a technical detail, we assume that any algorithm given (quantum) query access to a component mixer $\{M_i\}$ knows both n and the number of bits needed to encode an element of Ind_M.

We can now state the definitions of our query problems.

Definition 4. The SAME COMPONENT problem is: given query access to a component mixer $\{M_i\}$ on a set S and two elements $(s, t) \in S$, accept if s and t are in the same component of S.

Definition 5. The MULTIPLE COMPONENTS problem is: given query access to a component mixer $\{M_i\}$ on a partition $\{S_1, \ldots, S_c\}$, accept if $c > 1$.

Definition 6. The MULTIPLE BALANCED COMPONENTS problem is: There is a partition $\{S_1, \ldots, S_c\}$ with the promise that either there is only one component or no component contains more than half the elements in S. Given query access to a component mixer $\{M_i\}$ on that partition and the string 0^n, accept if $c > 1$.

On a quantum computer, we can also try to generate the uniform superposition over a component.

Definition 7. The COMPONENT SUPERPOSITION problem is: given quantum query access to a component mixer $\{M_i\}$ on a set S and an element $s \in S$, output the state

$$|S_j\rangle = \frac{1}{\sqrt{|S_j|}} \sum_{u \in S_j} |u\rangle$$

where S_j is the (unknown) component containing s.

The decision problems can also be viewed as complexity classes. We define the class SCP to be the set of languages that are polynomial-time reducible to the SAME COMPONENT problem with bounded error. Similarly, we define MCP by reference to MULTIPLE COMPONENTS and MBCP by reference to MULTIPLE BALANCED COMPONENTS.

3 Basic properties of component mixers

Lemma. (Component mixers are fully connected) *If s and t are in the same component, then there exists i such that $t = M_i(s)$.*

Proof. Assume the contrary. Suppose s and t are in the same component S_j and let $A = \{M_i(s) : i \in \mathrm{Ind}_M\}$. By assumption, $t \notin A$. This means that the variation distance between $M_i(s)$ (for $i \in_R \mathrm{Ind}_M$) and a uniform sample on S_j is

$$\begin{aligned}
&\frac{1}{2}\sum_{u\in S_j}\left|\Pr\left[M_i(s)=u\right]-\frac{1}{|S_j|}\right| \\
\geq&\frac{1}{2}\left|\Pr\left[M_i(s)=t\right]-\frac{1}{|S_j|}\right| \\
=&\frac{1}{2\,|S_j|} \\
\geq&2^{-n-1} > 2^{-n-2},
\end{aligned}$$

which contradicts the fact that M is a component mixer. □

A uniform quantum superposition over all of the elements in one component is a potentially useful state. It is not obvious whether a quantum computer can produce or verify such a state with a small number of queries to a component mixer, but it is possible to verify that a state is in the span of such superpositions.

Lemma. (Quantum computers can project onto component superpositions) *A quantum computer can, with a constant number of queries to a component mixer, measure the projector*

$$P = \sum_k \left(|S_k|^{-1/2}\sum_{x\in S_k}|x\rangle\right)\left(|S_k|^{-1/2}\sum_{x\in S_k}\langle x|\right) = \sum_k\left(\frac{1}{|S_k|}\sum_{x,y\in S_k}|x\rangle\langle y|\right)$$

with negligible error as a function of n.

Proof. Starting with a state $|\psi\rangle$, we give an algorithm to measure P on $|\psi\rangle$. The algorithm uses three registers: $|\psi\rangle$ starts in register A; registers B and C are ancillas. B's computational basis is Ind_M and C holds a single bit. To simplify the notation, we write the uniform superposition in register B as $|e_0\rangle = |\mathrm{Ind}_M|^{-1/2}\sum_i |i\rangle_M$. The algorithm is:

1. Initialize register B to $|e_0\rangle_B$ and register C to $|0\rangle$. This gives the state

$$|\phi_1\rangle = |\psi\rangle_A|e_0\rangle_B|0\rangle_C.$$

2. Apply controlled-M with the control set to 1. This is equivalent to applying M unconditionally. Let $\tilde{M}_j$ be the quantum operator corresponding to the action of M_j on register A. That is, $\langle s'|\tilde{M}_j|s\rangle = \langle s'|M_j(s)\rangle$. With this notation, the action of this step on registers A and B is $U = \left(\sum_j \tilde{M}_j \otimes |j\rangle\langle j|\right)$. The resulting state is

$$\begin{aligned}
|\phi_2\rangle &= U|\psi\rangle_A|e_0\rangle_B|0\rangle_C \\
&= |e_0\rangle_{BB}\langle e_0|U|\psi\rangle_A|e_0\rangle_B|0\rangle_C + \left(1-|e_0\rangle_{BB}\langle e_0|\right)U|\psi\rangle_A|e_0\rangle_B|0\rangle_C
\end{aligned}$$

3. Apply the unitary operator $|e_0\rangle_{BB}\langle e_0| \otimes X_C + (1 - |e_0\rangle_{BB}\langle e_0|) \otimes I_C$. This sets register C to $|1\rangle$ is register B is still in the state $|e_0\rangle$. The state is now

$$|\phi_3\rangle = |e_0\rangle_{BB}\langle e_0|U|\psi\rangle_A|e_0\rangle_B|1\rangle_C + (1 - |e_0\rangle_{BB}\langle e_0|)\, U|\psi\rangle_A|e_0\rangle_B|0\rangle_C.$$

4. Uncompute step 2 by applying $U^\dagger$. This gives

$$\begin{aligned}|\phi_4\rangle &= U^\dagger|e_0\rangle_{BB}\langle e_0|U|\psi\rangle_A|e_0\rangle_B|1\rangle_C + U^\dagger\,(1 - |e_0\rangle_{BB}\langle e_0|)\, U|\psi\rangle_A|e_0\rangle_B|0\rangle_C \\ &= \left(U^\dagger|e_0\rangle_{BB}\langle e_0|U\right)|\psi\rangle_A|e_0\rangle_B|1\rangle_C + \left(1 - U^\dagger|e_0\rangle_{BB}\langle e_0|U\right)|\psi\rangle_A|e_0\rangle_B|0\rangle_C.\end{aligned}$$

To simplify this result, observe that the matrix $|\mathrm{Ind}_M|^{-1}\sum_j \tilde{M}_j$ is the Markov matrix obtained by applying one of the M_i uniformly at random to an element of S. From the definition of a component mixer, $|\mathrm{Ind}_M|^{-1}\sum_j \tilde{M}_j \approx P$. Furthermore, $P|\psi\rangle$ has the form $\alpha\sum_{x\in S_a}|x\rangle$ for some a and α, and M_k preserves the set S_a, so $\tilde{M}_k P|\psi\rangle = P|\psi\rangle$ for all k. Using these observations, we can simplify

$$\begin{aligned}U^\dagger|e_0\rangle_{BB}\langle e_0|U|\psi\rangle_A|e_0\rangle_B &= \left(\sum_k \tilde{M}_k^\dagger \otimes |k\rangle_{BB}\langle k|\right)|e_0\rangle_{BB}\langle e_0|\left(\sum_j \tilde{M}_j \otimes |j\rangle\langle j|\right)|\psi\rangle_A|e_0\rangle_B \\ &= \left(\sum_k \tilde{M}_k^\dagger \otimes |k\rangle_{BB}\langle k|\right)|e_0\rangle_B\,|\mathrm{Ind}_M|^{-1}\left(\sum_j \tilde{M}_j\right)|\psi\rangle_A \\ &\approx \left(\sum_k \tilde{M}_k^\dagger \otimes |k\rangle_{BB}\langle k|\right)P|\psi\rangle_A|e_0\rangle_B \\ &= P\left(\sum_k |k\rangle_{BB}\langle k|\right)|\psi\rangle_A|e_0\rangle_B \\ &= P|\psi\rangle_A|e_0\rangle_B.\end{aligned}$$

Plugging this in, we have

$$|\phi_4\rangle \approx P|\psi\rangle_A|e_0\rangle_B|1\rangle_C + (1 - P)\,|\psi\rangle_A|e_0\rangle_B|0\rangle_C$$

with negligible error.

At this point, register B is unentangled with the rest of the system, register C contains the outcome of the measurement we wanted, and register A contains the correct final state. □

On a quantum computer, SAME COMPONENT reduces to COMPONENT SUPERPOSITION: given two initial elements, a swap test can decide with bounded error whether their respective component superpositions are the same state or non-overlapping states.

4 Placing component mixer problems in the complexity zoo

4.1 Inclusions

Several of the complexity class relationships in Table 1 are straightforward. MULTIPLE COMPONENTS is a relaxation of MULTIPLE BALANCED COMPONENTS, so MBCP ⊆ MCP. The "component mixers are fully connected" lemma implies a simple NP algorithm for SAME COMPONENT, so SCP ⊆ NP. MULTIPLE COMPONENTS can be restated as "do there exist two objects that are *not* in the same component?", so MCP ⊆ $\mathrm{NP}^{\text{co-SCP}}$ and hence MCP ⊆ $\mathrm{NP}^{\text{co-NP}}$.

In the appendix, we give two Arthur-Merlin protocols for MULTIPLE BALANCED COMPONENTS:

- A protocol to prove a "yes" answer. In this protocol, Merlin solves the SAME COMPONENT problem on input given by Arthur (appendix A.1).
- A protocol to prove a "no" answer (appendix A.2).

The existence of these protocols implies that MBCP $\subseteq$ AM, $\mathrm{BPP}^{\mathrm{SCP}}$, and co-AM. We also give a QMA protocol for MULTIPLE COMPONENTS (see appendix A.3).

SAME COMPONENT is reducible to STATISTICAL DIFFERENCE: to test whether s and t are in the same component, choose $i, j \in_R \mathrm{Ind}_M$ and test whether $M_i(s)$ and $M_j(t)$ have the same distribution. STATISTICAL DIFFERENCE is complete for SZK, so SCP $\subseteq$ SZK [7].

MULTIPLE BALANCED COMPONENTS also reduces to STATISTICAL DIFFERENCE: choose $a, b \in_R S$ and $i, j \in_R \mathrm{Ind}_M$. If there are multiple balanced components, then the predicate that the first two and last two elements of $(a, M_i(a), b, M_j(b))$ are in the same component holds w.p. 1, whereas the same predicate holds on four independent uniform samples from S w.p. at most $1/4$. This means that the variation distance between $(a, M_i(a), b, M_j(b))$ and four independent samples is at least $3/4$. If, on the other hand, there is only one component, then $(a, M_i(a), b, M_j(b))$ is negligibly different four independent samples from S. Therefore, MULTIPLE BALANCED COMPONENTS reduces to STATISTICAL DIFFERENCE on the distribution of $(a, M_i(a), b, M_j(b))$ versus four independent uniform samples from S. Hence MBCP $\subseteq$ SZK.

4.2 Separations

SCP contains group membership (relative to any oracle) and group membership is not in co-MA for black-box groups [3], so SCP $\not\subseteq$ co-MA relative to an oracle.

The quantum query complexity of MULTIPLE COMPONENTS is exponential by reduction from the Grover problem (see appendix B). This implies the existence of an oracle separating MCP and BQP.

4.3 Conjectured separations

We conjecture that there is no QCMA or co-QCMA proof for MULTIPLE COMPONENTS or even MULTIPLE BALANCED COMPONENTS, which would imply the existence of an oracle separating MBCP from QMA and hence QCMA from QMA.

We further conjecture that MULTIPLE BALANCED COMPONENTS has superpolynomial randomized and quantum query complexity. This conjecture would imply that MBCP is separated from BPP and BQP by an oracle.

5 A hardness result for counterfeiting quantum money

We are now ready to prove a hardness result for counterfeiting quantum money. Recall that the quantum money state is defined [2, 1] as

$$|\$_\ell\rangle = \sum_{x \in S_\ell} |x\rangle$$

where S_ℓ is a component of a partition of a big set S and an adversary is given access to a component mixer for that partition. Unlike the other component mixer problems we have discussed, an adversary also has access to a labeling function L that maps each element of S to a label that identifies which component that element is in.

We show that, if an attacker is given one copy of $|\$_\ell\rangle$ and *measures* it in the computational basis, then, under reasonable assumptions, the attacker cannot recreate the state. That is, given some

$s \in S_\ell$ (i.e. the measurement outcome), it is hard to produce $|\$_\ell\rangle$. We call this type of attack SIMPLE COUNTERFEITING. Our assumption is that the quantum query complexity of SAME COMPONENT is superpolynomial.

Definition 8. The SIMPLE COUNTERFEITING problem is: given quantum query access to a component mixer $\{M_i\}$ on a set S, quantum query access to a function L that maps each element of S to a unique label identifying the component containing that element, and an element $s \in S$, output the state

$$|S_j\rangle = \frac{1}{\sqrt{|S_j|}} \sum_{u \in S_j} |u\rangle$$

where S_j is the component containing s.

SIMPLE COUNTERFEITING is the same problem as COMPONENT SUPERPOSITION except that the algorithm also has access to the labeling function. This makes the problem seem easier; for example, SAME COMPONENT and MULTIPLE BALANCED COMPONENTS both become trivial with access to the labeling function. We show that the labeling function is unhelpful for the purpose of SIMPLE COUNTERFEITING.

Theorem. *If the quantum query complexity of* COMPONENT SUPERPOSITION *is superpolynomial, then the quantum query complexity of* SIMPLE COUNTERFEITING *is also superpolynomial.*

Proof. The SIMPLE COUNTERFEITING and COMPONENT SUPERPOSITION problems differ in that SIMPLE COUNTERFEITING is given access to a label that identifies components. Calculating such a label given only a component mixer is at least as hard as solving SIMPLE COUNTERFEITING in the first place, so we won't be able to provide a valid label. The idea behind the proof is to show that a correct labeling function is not very helpful for solving SIMPLE COUNTERFEITING, and that, given a component mixer, we can efficiently provide a label that is indistinguishable from a valid label in polynomial time.

We assume for contradiction that we have a quantum query algorithm "alg" that solves SIMPLE COUNTERFEITING in n^k queries for sufficiently large n. Alg is given quantum query access to a component mixer and labeling function and it is promised that the labeling function is consistent with the component mixer. It takes as input an element $s \in S_j$ for some j. It makes n^k quantum queries and produces a mixed state ρ as output. The trace distance between ρ and the desired output state $\frac{1}{\sqrt{|S_j|}} \sum_{u \in S_j} |u\rangle$ is a negligible function of n.

We give an algorithm that solves COMPONENT SUPERPOSITION with high probability using alg as a subroutine.

As input, we have quantum query access to a component mixer on n bits and an n-bit string s. This means that the space of n bit strings is partitioned into components $S_1, \ldots, S_c$ and a set of "garbage" strings $G = \{0,1\}^n \setminus (S_1 \cup \cdots \cup S_c)$, where c is the (unknown) number of components. We are not given access to a labeling function. WLOG, we assume that $s \in S_1$.

We define an instance of SIMPLE COUNTERFEITING on $2n$-bit strings that can be used to solve the original COMPONENT SUPERPOSITION problem. To simplify the notation, we treat each $2n$-bit string as a pair of binary numbers, each between 0 and $2^n - 1$. In our instance of SIMPLE COUNTERFEITING, the components are $\{0\} \times S_1, \ldots, \{0\} \times S_c$ and $\{0\} \times G$. Each other element (that is, everything that has something nonzero as its first n bits) is its own component. We use the component mixer

$$M_i^{(0)}(r,z) = \begin{cases} (0, M_i(z)) & \text{if } r = 0 \\ (r,z) & \text{otherwise} \end{cases}$$

and *incorrect* label

$$L^{(0)}(r,z) = \begin{cases} (0,0) & \text{if } r = 0 \\ (r,z) & \text{otherwise} \end{cases}.$$

The label $L^{(0)}$ violates the promise of SIMPLE COUNTERFEITING (it assigns the same label to all of the components in the original component mixer), so the SIMPLE COUNTERFEITING algorithm run directly on $\left\{M_i^{(0)}\right\}$ and $L^{(0)}$ might fail. However, the only way to detect that $L^{(0)}$ is invalid is to query it on some input of the form $(0,t)$ for $t \in S_2 \cup \cdots \cup S_c$. Those inputs are an exponentially small fraction of the domain of $L^{(0)}$ and we can hide them by randomly permuting $L^{(0)}$ and $M_i^{(0)}$, giving this algorithm:

1. Choose independent random permutations π and σ on $\mathbb{Z}_{2^n} \times \mathbb{Z}_{2^n}$. π indicates where each $2n$-bit string is hidden in the permuted problem and σ scrambles the labels. (These permutations will take an exponential number of bits to specify, but they can be implemented with no queries to $\{M_i\}$.)
2. Run alg on $\left\{\pi \circ M_i^{(0)} \circ \pi^{-1}\right\}$ and $\sigma \circ L^{(0)} \circ \pi^{-1}$ with the initial element $\pi(0,s)$.
3. Apply π^{-1} coherently to the quantum state that alg produces.
4. Output the last n qubits of the result.

If $\sigma \circ L^{(0)} \circ \pi$ were a valid label function for the component mixer $\left\{\pi \circ M_i^{(0)} \circ \pi\right\}$, then this algorithm would succeed on each try w.p. negligibly different from 1. We will prove that the invalidity of the labeling function is well enough hidden that the algorithm works anyway.

To prove this, we assume the contrary: there is some $\{M_i\}$ for which this algorithm fails with nonnegligible probability. This means that the actual output of our algorithm differs nonnegligibly in trace distance from the desired output. Such a difference would be detectable if we knew what the correct output was; we will show that this is impossible by solving the Grover problem more quickly than is allowed by the BBBV theorem using alg as a subroutine.

We generalize the functions $M_i^{(0)}$ and $L^{(0)}$ to a larger family that encodes a Grover search problem. We can picture $\{M_i^{(0)}\}$ as an embedding of the original problem in the first row of a grid in which the first n bits is the row index and the last n bits is the column index (see Figure 1—the unmarked squares are their own components). There are many other ways we could have embedded the original problem, though. (These other embeddings are well-defined, but they are difficult to calculate without access to a labeling function for the original problem.) In particular, we could have placed everything except S_1 on a different row. If we put the other components on the j^{th} row, we get

$$L^{(j)}(r,z) = \begin{cases} (0,0) & \text{if } r = 0 \text{ and } z \in S_1 \\ (0,0) & \text{if } r = j \text{ and } z \notin S_1 \\ (r,z) & \text{otherwise} \end{cases}$$

and

$$M_i^{(j)}(r,z) = \begin{cases} (0, M_i(z)) & \text{if } r = 0 \text{ and } z \in S_1 \\ (j, M_i(z)) & \text{if } r = j \text{ and } z \notin S_1 \\ (r,z) & \text{otherwise} \end{cases}.$$

Alternatively, we could leave them out entirely, giving

$$L^{\text{nowhere}}(r,z) = \begin{cases} (0,0) & \text{if } r = 0 \text{ and } z \in S_1 \\ (r,z) & \text{otherwise} \end{cases}$$

and

$$M_i^{\text{nowhere}}(r,z) = \begin{cases} (0, M_i(z)) & \text{if } r = 0 \text{ and } z \in S_1 \\ (r,z) & \text{otherwise} \end{cases}.$$

We can't efficiently implement queries to L^{nowhere}, M^{nowhere}, $L^{(j)}$ or $M_i^{(j)}$ for $j \neq 0$, but, if we could and if SIMPLE COUNTERFEITING didn't notice that the label function was invalid, then the output on any of instances with starting element $(0,s)$ would be

$$\sum_{z \in S_1} |0\rangle|z\rangle,$$

the latter n qubits of which is exactly the state we wanted.

The function L^{nowhere} is a valid labeling function, but all of the $L^{(j)}$ are invalid because they take the same value on the images of $S_1, \ldots, S_c$ even though they are in different components. Nonetheless, they look valid as long as no one ever queries them on the images of $S_2, \ldots, S_c$, which collectively represent less than a 2^{-n} fraction of all possible queries.

We formalize this notion by a reduction from the Grover problem. Suppose $g : \mathbb{Z}_{2^n} \to \{0,1\}$ is a function that outputs 1 at most one input. By the BBBV theorem [8], the query complexity of distinguishing a random point function g from all zeros is $O\left(2^{n/2}\right)$. Using our algorithm for SIMPLE COUNTERFEITING as a subroutine, we will attempt to decide whether g maps any value to 1. We do this by allowing g to select which embedding to use. This gives the "labeling" function

$$L^{[g]}(r,z) = \begin{cases} (0,0) & \text{if } r = 0 \text{ and } z \in S_1 \\ (0,0) & \text{if } g(r) = 1 \text{ and } z \notin S_1 \\ (r,z) & \text{otherwise} \end{cases}$$

and component mixer

$$M_i^{[g]}((r,z)) = \begin{cases} (0, M_i(z)) & \text{if } r = 0 \text{ and } z \in S_1 \\ (j, M_i(z)) & \text{if } g(r) = 1 \text{ and } z \notin S_1 \\ (r,z) & \text{otherwise} \end{cases}.$$

If $g(j) = 1$ for some j, then $L^{[g]} = L^{(j)}$ and $M_i^{[g]} = M_i^{(j)}$; otherwise $L^{[g]} = L^{\text{nowhere}}$ and $M^{[g]} = M^{\text{nowhere}}$. It is possible to evaluate either $L^{[g]}(r,z)$ or $M_i^{[g]}(r,z)$ with a very large number of queries to the original component mixer $\{M_i\}$ and *one* query to $g(r)$. (Evaluating the functions coherently requires a second query to $g(r)$ to uncompute garbage.)

If we choose independent random permutations π and σ on $\mathbb{Z}_{2^n} \times \mathbb{Z}_{2^n}$ and run alg on $\left\{\pi \circ M_i^{[g]} \circ \pi^{-1}\right\}$ and $\sigma \circ L^{[g]} \circ \pi - 1$ with initial state $\pi(0,s)$, the output of the algorithm is some mixed state that depends on g. Let ρ_0 be the density matrix of that mixed state if g is all zeros and let ρ_{point} be the density matrix if g is a uniformly random point function.

Claim. $\|\rho_0 - \rho_{\text{point}}\|_{\text{tr}}$ is a negligible function of n.

Proof. Assume the contrary: $\|\rho_0 - \rho_{\text{point}}\|_{\text{tr}} \geq n^{-k}$ for some fixed k and an infinite sequence of values of n. If we run alg on $L^{[g]}$ and $M_i^{[g]}$, we can then decide whether the output is ρ_0 or ρ_{point}

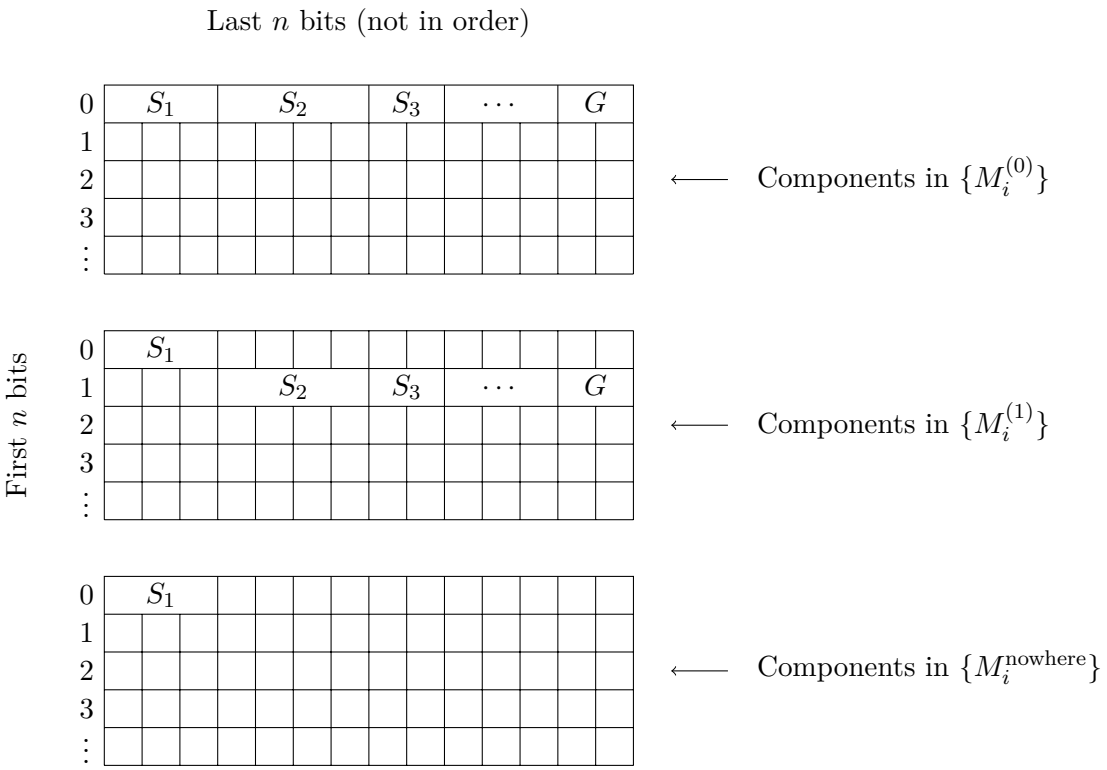

Figure 1: There are $2^n + 1$ ways to hide components that are hard to label.

and therefore whether g is all zeros or a point function by measuring the output. We will get the right answer w.p. at least $\frac{1}{2} + \frac{n^{-k}}{2}$. We can amplify n^{2k} times to get the right answer w.p. at least $2/3$ by a Chernoff bound. By assumption, alg makes n^r queries to $L^{[g]}$ and $M_i^{[g]}$. That means that, in $n^{r+2k} = o\left(n^{n/2}\right)$ queries, we can determine whether $g(j) = 1$ for any j, which is impossible by the BBBV theorem. Therefore $\|\rho_0 - \rho_{\text{point}}\|_{\text{tr}}$ is a negligible function of n. □

It follows that, if we apply π^{-1} to ρ_0 and to ρ_{point}, the results differ negligibly in trace distance. The result of applying π^{-1} to ρ_0 is the uniform superposition over $\{0\} \times S_1$ up to negligible error because if $g = 0$ then alg's promise is satisfied and it produces the correct answer. Furthermore, if we set $g(0) = 1$, then the output distribution is still ρ_0 because the distribution of component mixers and labels seen by alg is independent of which point function we choose. This means that π^{-1} applied to the output of alg on $\left\{\pi \circ M_i^{(0)} \circ \pi^{-1}\right\}$ and $\sigma \circ L^{(0)} \circ \pi^{-1}$ with initial state $\pi(0, s)$ differs negligibly from the uniform superposition over $\{0\} \times S_1$ in trace distance.

This contradicts the assumption that there exists some input on which our algorithm fails, so our algorithm solves COMPONENT SUPERPOSITION with negligible error. □

We can replace the assumption that COMPONENT SUPERPOSITION is hard with the assumption that SAME COMPONENT is hard because SAME COMPONENT reduces to COMPONENT SUPERPOSITION.

6 Open problems

There are a number of open problems related to this work.

Ideally, we would prove the impossibility of more general forms of counterfeiting. If we could show that, given one copy of $|\$_\ell\rangle$ for some ℓ, it is hard to produce a second copy of $|\$_\ell\rangle$, then we would know that (in a black-box model) quantum money could not be counterfeited. An even better result would be collision-freedom: that is hard for anyone to produce a state of the form $|\$_\ell\rangle \otimes |\$_\ell\rangle$ by any means, even for a random ℓ of an attacker's choice. (Collision-freedom implies that copying is impossible: if an attacker could copy a given quantum money state, then the output of the algorithm would be contain two copies of $|\$_\ell\rangle$ for the value of ℓ implied by the input.)

It should be possible to prove quantum lower bounds on the query complexity of SAME COMPONENT and MULTIPLE BALANCED COMPONENTS. This would strengthen the hardness result for counterfeiting quantum money.

A classical oracle separating MCP and QCMA would also separate QMA and QCMA. We conjecture that an appropriate worst-case component mixer would work, but we have no proof.

A cryptographically secure component mixer could be a useful object, and a good cryptographically secure component mixer with an associated labeling function would give a better quantum money protocol than quantum money from knots. (Knot invariants have all kinds of unnecessary properties.) If we had that as well as a hardness result for generating quantum money collisions, then quantum money would be on a sound theoretical footing.

7 Acknowledgments

I would like to thank Eddie Farhi for extensive comments, Scott Aaronson for suggestions about complexity classes, Peter Shor and Avinatan Hassidim for encouraging me to look for reductions from problems like graph isomorphism to counterfeiting quantum money, and Jon Kelner for valuable thoughts about component mixers.

This work was supported by the Department of Defense (DoD) through the National Defense Science & Engineering Graduate Fellowship (NDSEG) Program as well as the U.S. Department of Energy under cooperative research agreement No. DE-FG02-94ER40818.

References

[1] E. Farhi, D. Gosset, A. Hassidim, A. Lutomirski, and P. Shor, "Quantum money from knots," `arXiv:1004.5127 [quant-ph]`.

[2] A. Lutomirski, S. Aaronson, E. Farhi, D. G osset, A. Hassidim, J. Kelner, and P. Shor, "Breaking and making quantum money: toward a new quantum cryptographic protocol," in *Innovations in Computer Science*. 2010. `arXiv:0219.3825 [quant-ph]`. `http://conference.itcs.tsinghua.edu.cn/ICS2010/content/papers/2.html`.

[3] J. Watrous, "Succinct quantum proofs for properties of finite groups," *Foundations of Computer Science, Annual IEEE Symposium on* (2000) 537–456.

[4] S. Aaronson and G. Kuperberg, "Quantum versus classical proofs and advice," *Theory of Computing* **3** no. 1, (2007) 129–157.

[5] L. Babai, "Local expansion of vertex-transitive graphs and random generation in finite groups," in *Proceedings of the twenty-third annual ACM symposium on Theory of computing*, STOC '91, pp. 164–174. ACM, New York, NY, USA, 1991.

[6] L. Grover and T. Rudolph, "Creating superpositions that correspond to efficiently integrable probability distributions," `arXiv:quant-ph/0208112`.

[7] A. Sahai and S. Vadhan, "A complete problem for statistical zero knowledge," *J. ACM* **50** (March, 2003) 196–249.

[8] C. H. Bennett, E. Bernstein, G. Brassard, and U. Vazirani, "Strengths and weaknesses of quantum computing," *SIAM J. Comput.* **26** (October, 1997) 1510–1523.

A Query protocols for component problems

A.1 An AM query protocol multiple balanced components

Suppose that Merlin wants to prove to Arthur that some component mixer has multiple balanced components. Arthur and Merlin run this protocol:

	Arthur	Merlin
1.	Choose $s_1, s_2 \in_R S$, $i \in_R \{1,2\}$ and $j \in_R \mathrm{Ind}_M$.	
2.	Compute $t = M_j(s_i)$.	
3.	Send s_1, s_2, t to Merlin.	
4.		If s_1 and s_2 are in different components, compute $i' = i$. Otherwise, choose $i' \in_R \{1,2\}$.
5.		Send i' to Arthur.
6.	Accept iff $i = i'$.	

If $\{M_i\}$ has multiple balanced components, then with probability at least $1/2$, s_1 and s_2 are in different components. In this case, Merlin will always answer correctly. This means that Merlin is correct w.p. at least $3/4$. If, on the other hand, M has only one component, then t is a nearly uniform sample from S (trace distance at most $2^{-n-2} \leq 1/8$). This means that Merlin can guess i correctly with probability at most $5/8$. With constant overhead, this protocol can be amplified to give soundness and completeness errors $1/3$.

Steps 1, 2, 3, 5, and 6 can be done in a constant number of queries to the component mixer oracle. Step 4 requires Merlin to solve the SAME COMPONENT to decide whether t is in the same component as s_1, s_2, or both. This means that if Arthur had the power of SCP (with oracle access to $\{M_i\}$), then he could run the protocol on his own.

A.2 A co-AM query protocol for multiple balanced components

Suppose that Merlin wants to prove to Arthur that some component mixer has a single component (as opposed to multiple balanced components). Arthur and Merlin run this protocol:

	Arthur	Merlin
1.	Choose $s_1, s_2 \in_R S$.	
2.	Send s_1, s_2 to Merlin.	
3.		Choose $i \in_R \mathrm{Ind}_M$ such that $M_i(s_1) = s_2$.
4.		Send i to Arthur.
5.	Accept iff $M_i(s_1) = s_2$.	

If there is only one component, then s_1 and s_2 are in the same component and Merlin can find i because component mixers are fully connected. If, on the other hand, there are multiple balanced components, then w.p. at least $1/2$, s_1 and s_2 are in different components and no such i exists.

This means that this proof is complete and has soundness error at most $1/2$. A constant amount of amplification will reduce the soundness error below $1/3$.

A.3 A quantum witness for multiple components

Given a "yes" instance of MULTIPLE COMPONENTS problem, let S_1 and S_2 be two distinct components. Then a valid witness state is

$$|\psi_{\mathrm{MC}}\rangle = \left(\sum_{s\in S_1} |s\rangle\right) \otimes \left(\sum_{s\in S_2} |s\rangle\right).$$

To verify the witness, Arthur first measures the projector of each register onto the space of uniform superpositions over components (see section 3). If either measurement outputs zero, Arthur rejects. Otherwise Arthur performs a swap test between the two registers and accepts iff the swap test says that the registers are different.

On a valid witness, Arthur's projections succeed with probability close to 1. The states in the two registers have disjoint support (both before and after the swap test), so the swap test indicates that the states are different w.p. $1/2$. Arthur therefore accepts a valid witness w.p. $1/2$.

If there is only one component then projecting onto the space of uniform superpositions over components is equivalent to projecting onto the uniform superposition over S. Therefore, on any witness, if Arthur's projections succeed then the post-measurement state is (up to negligible error) two copies of the uniform superposition over S. Those two copies are approximately the same state, so the swap test says that they are the same and Arthur rejects w.p. near 1. Standard techniques can amplify this protocol to give completeness and soundness errors less than $1/3$.

B MULTIPLE COMPONENTS has exponential quantum query complexity

We can embed an instance of the Grover problem into MULTIPLE COMPONENTS. Let g be the instance of the Grover problem on n bits (i.e. $g : \mathbb{Z}_{2^n} \to \{0,1\}$ is either all zeros or a point function). Let $\mathrm{Ind}_M = \mathbb{Z}_{2^n}$ and define the component mixer

$$M_i(x) = \begin{cases} (x+i) \bmod 2^n & \text{if } g(x) = g(x+i) = 0 \\ x & \text{otherwise} \end{cases}.$$

If g is all zeros then there is a single component but if $g(y) = 1$ then y is in its own component. The function M_i can be evaluated with two queries to g, so the Grover decision problem on g reduces to MULTIPLE COMPONENTS on $\{M_i\}$.

Hence, by the BBBV theorem [8], the quantum query complexity of MULTIPLE COMPONENTS is $\Omega\left(2^{n/2}\right)$.